Podstawy Sieci Neuronowych

Wroclaw University of Science and Technology

Wydział Informatyki i Telekomunikacji

Ziarnko nadziei

Rozpoznawanie ziarenek ryżu

18 grudnia 2024

Prowadzący kurs: Dr Inż. Aneta Górniak
Autorzy: Maria Raczkiewicz, Maksymilian Wiśniewski
Zajęcia: Poniedziałek 17:05
Github: Ziarnko nadziei

1 Wstęp

Celem projektu było zaprojektowanie, zaimplementowanie i wytrenowanie sztucznej sieci neuronowej, zdolnej do rozwiązania średniozaawansowanego problemu klasyfikacyjnego – rozpoznawania gatunków ryżu na podstawie obrazu. Realizacja projektu odbyła się w środowisku MATLAB, z wykorzystaniem narzędzi dostępnych w Deep Learning Toolbox. Do treningu i testowania modelu wykorzystano gotowy zbiór danych dostępny pod adresem: Rice Image Dataset.

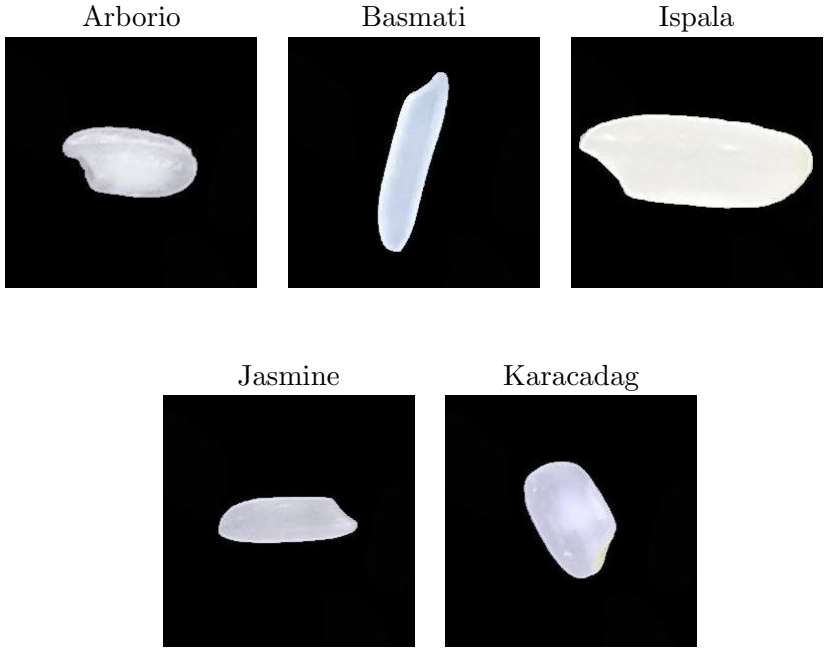


Tabela 1: Przykładowe obrazy z bazy

2 Przygotowania

Analiza problemu i architektura
Obrazy wybranej bazy danych są bardzo czytelne dla sieci neuronowej, a zarazem ich zbiór jest stosunkowo duży bo mieszczący 15 tysięcy obrazów na kategorie. W związku z bazą obrazów początkową architekturą była CNN (Convolutional Neural Network), jednak z czasem architektura uległa zmianie na MLP (Multilayer Perceptron), o czym w dalszej części.

Podział danych
Zbiór danych został podzielony na 4 kategorie:

Training	Validation	Testing	Play
70%	14%	15%	1%

Tabela 2: Podział danych

- Training - jest zbiorem wykorzystywanym do trenowania sieci i powinien być najliczniejszy
- Validation - odpowiada za weryfikacjie czy sieć nie dopasowała się nadmiernie do Training setu.

- Testing - zbiór służący do weryfikacji jakości z jaką nauczyła się sieć
- Play - mały zbiór stworzony do ewentulanego ręcznego testowania sieci na pojedynczych przykładach

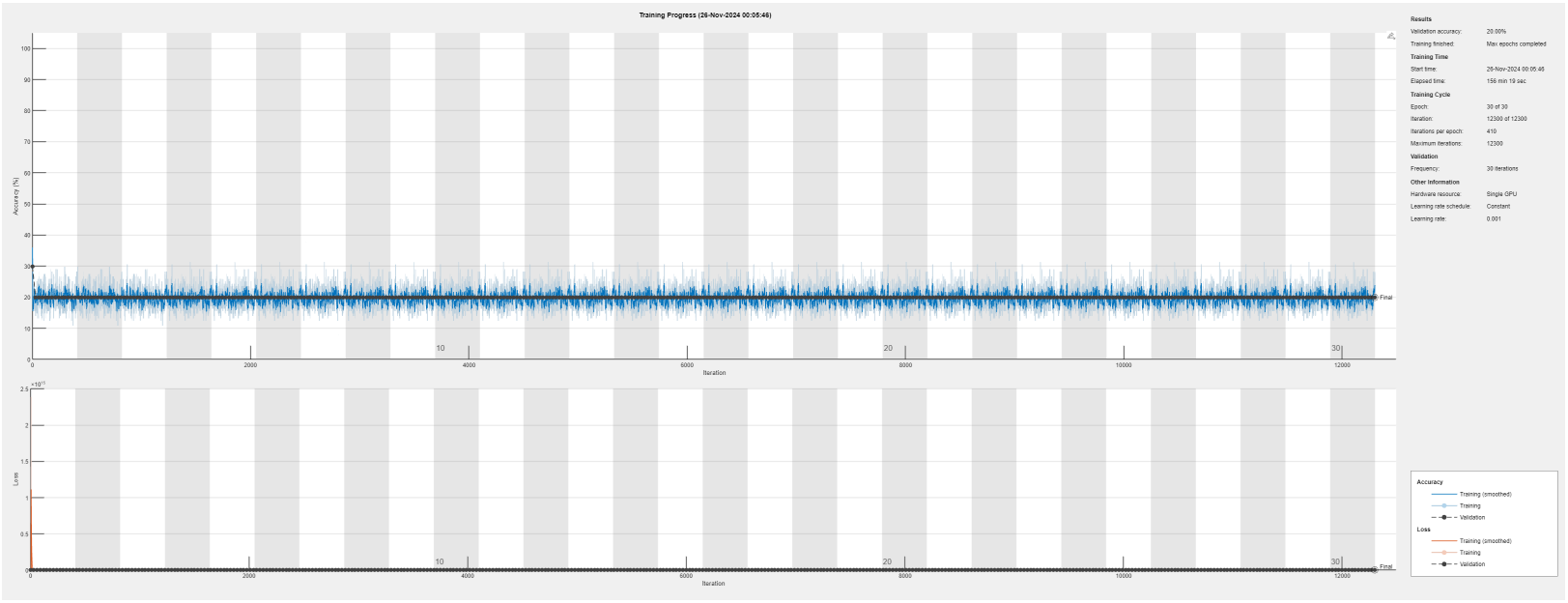
Przygotowanie danych
Obrazy same w sobie są bardzo korzystne dla sieci dzięki: stosunkowo wysokiej rozdzielczości (250x250), jednolitym kontrastowym tle oraz formacie w skali szarości. Gatunki różnią się głównie kształtem oraz stopniem przeźroczystości. Zdjęcia ponadto zostały wykonane w różnych orientacjach ziarenek z pozwoliło na większą uniwersalność sieci oraz wyklucza konieczność sztucznego obracania obrazem.

Baza została w pełni pobrana, następnie za pomocą metody `imageDatastore()` wprowadzona do programu zgodnie z podziałem na foldery które dzielą bazę na poszczególne gatunki. `categories(.Labels)` przydzieliło odpowiednie etykiety obrazom w folderach zgodnie z ich gatunkiem. `splitEachLabel()` podzieliło bazę na podzbiory w odpowiednich proporcjach oraz je przetasowało. Na koniec wykorzystując metody `im2gray()` pozbyto się z obrazów zbędnych pusty warstw kolorów w celu usprawnienia pracy sieci.

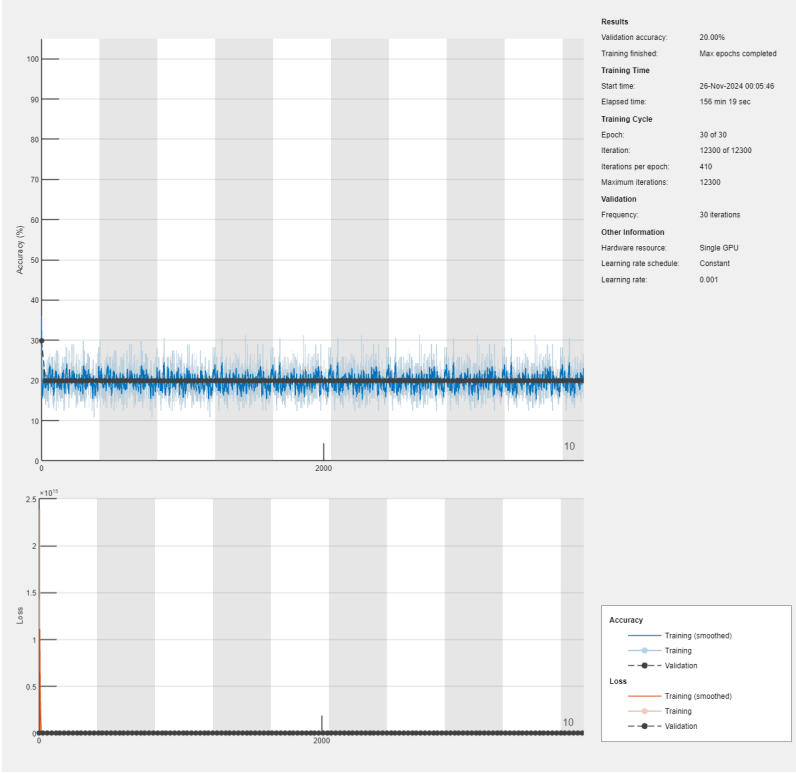
3 Badania

Domyślnie (chyba że opisano inaczej) używano optymalizatora *sgdm* (Stochastic Gradient Descent with Momentum) oraz funkcję aktywacji ReLU, której ilość była odpowiednio dostosowywana do zmieniającej się liczby warstw. Warstwa konwolucyjna zawsze z akompaniamentem maxPoolingLayer oraz z ustawieniami: *convolution2dLayer(3, 32, "Padding", 0)* i *maxPooling2dLayer(5, 'Stride', 3)* Jako zasób sprzętowy do trenowania sieci wybrano kartę graficzną, korzystano z modelu NIVIDA GeForce RTX 3070.

Początkowo sieć była testowana dość eksperymentalnie i nieefektywnie, zdecydowano się na dużą ilość epok - 30, wysoki (w kontekście badanej sieci) współczynnik uczenia - 0.001 (Ilr), jedną warstwę konwolucyjną (Conv), dwie warstw w pełni połączone (Fcl).



Rysunek 1: Wykres dla Ilr=0.001, 1xConv i 2xFcl(256 i 5).

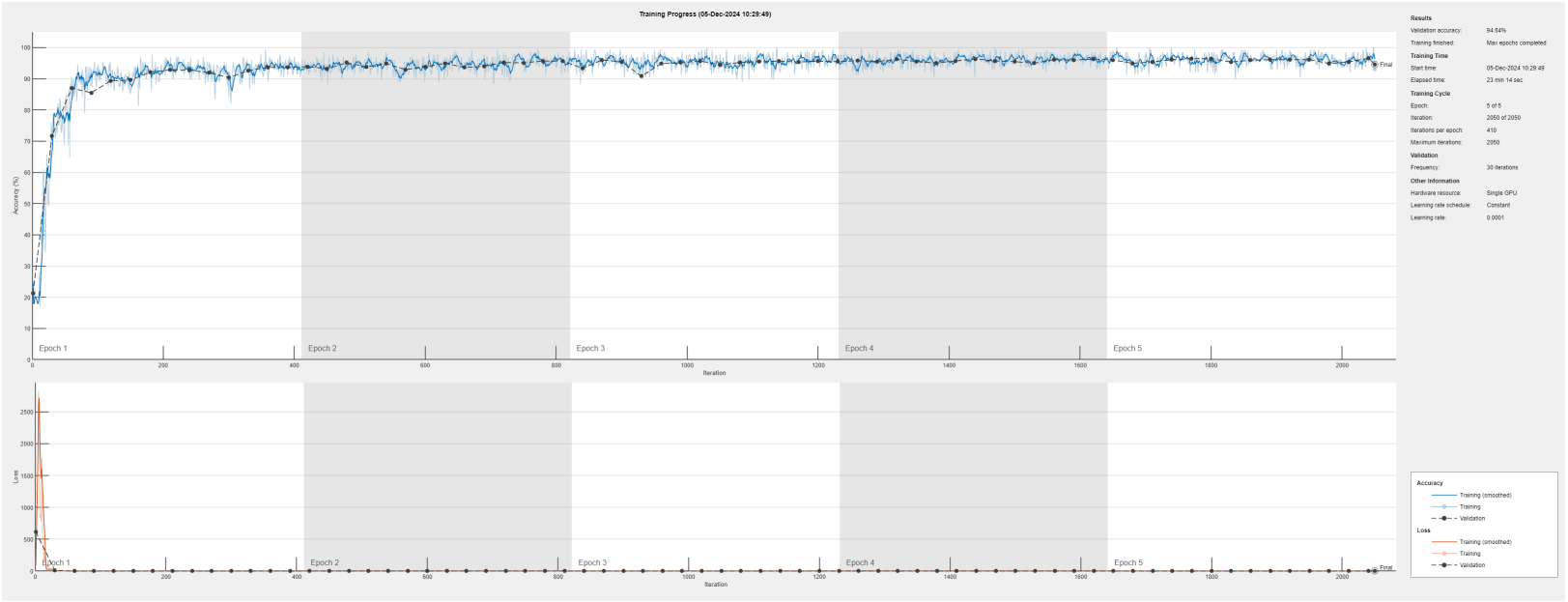


Rysunek 2: Powiększenie Rysunku 1.

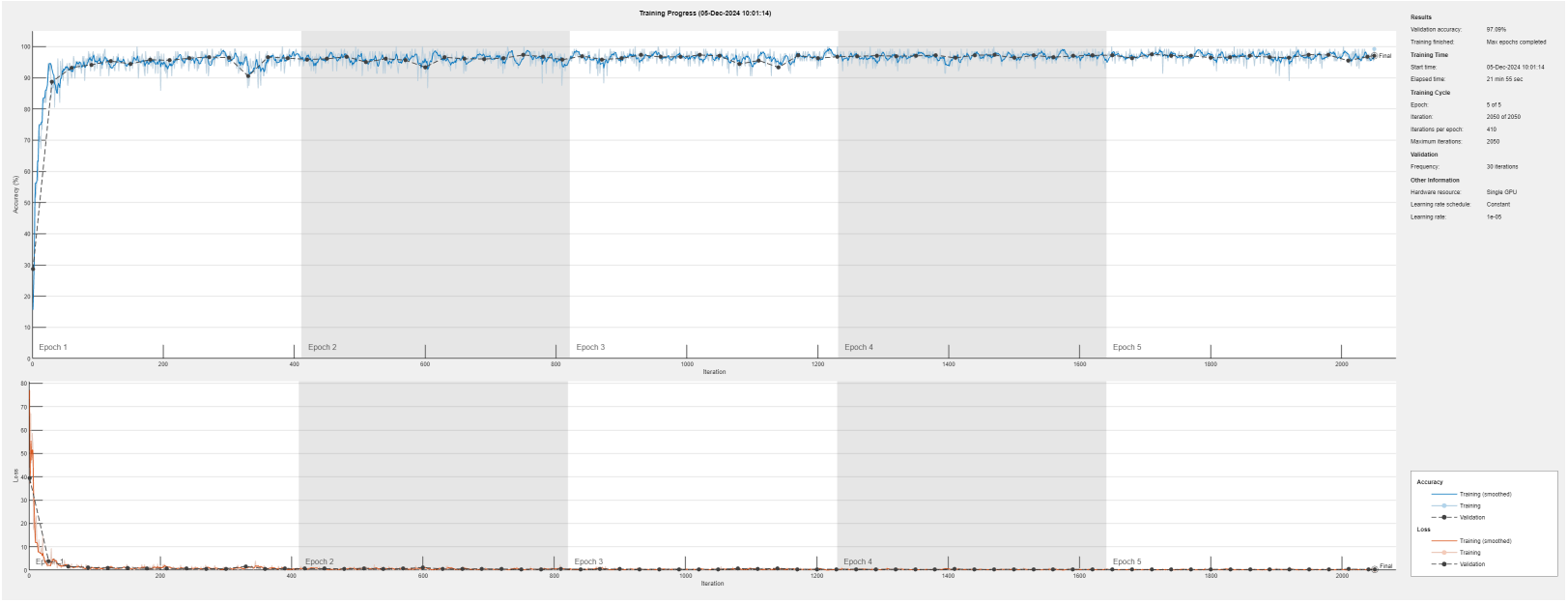
Na przedstawionych wykresach można zauważyć, że sieć neuronowa nie uczyła się - dokładność utrzymywała się na stałym poziomie 20%, a charakterystyka wykresu powtarzała się cyklicznie w każdej epoce. Przyczyną tego problemu był zbyt wysoki współczynnik uczenia się, który uniemożliwił

modelowi skuteczną aktualizację wag.

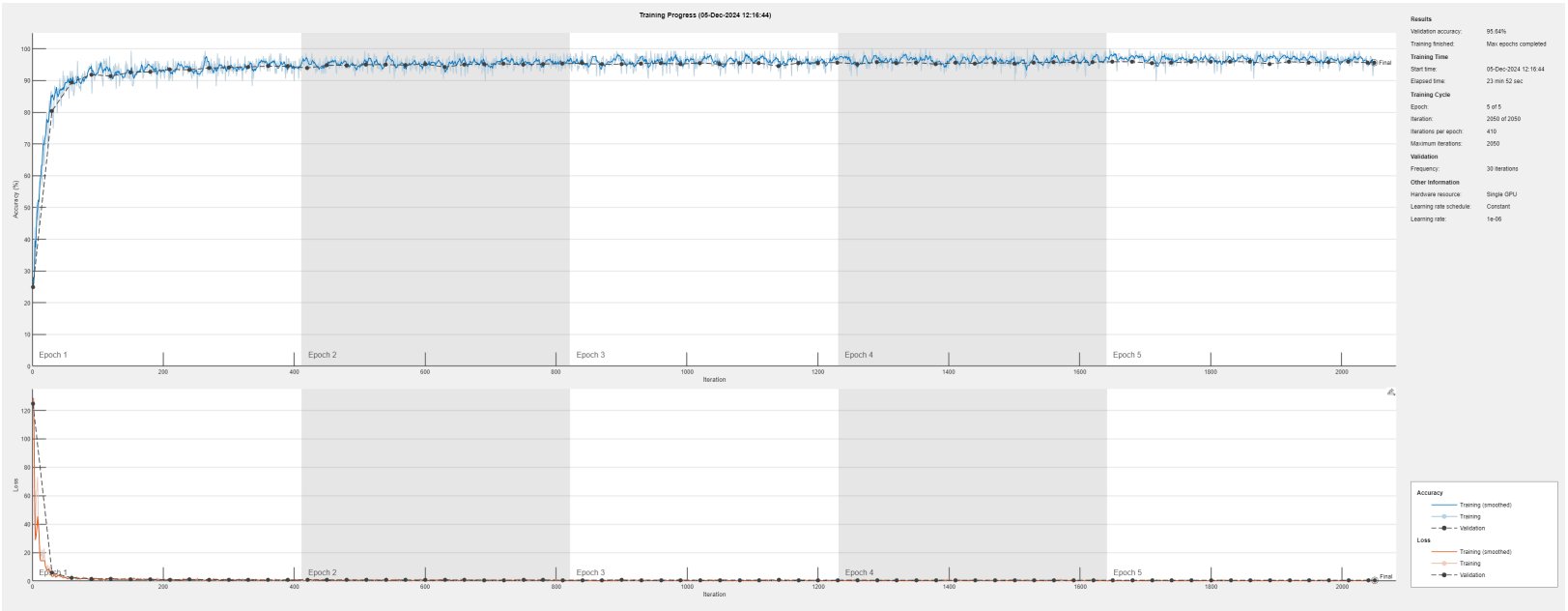
W następnych testach zrezygnowano z architektury CNN i skupiono się jedynie na MLP, w celu znalezienia najlepszego współczynnika uczenia. Badania na poniższych obrazach przeprowadzano na trzech w pełni połączonych warstwach o liczbie neuronów ukrytych odpowiednio wynoszących 512, 256 oraz 5, jedynym modyfikowanym parametrem był współczynnik uczenia. W opisie obrazu przedstawiono dokładność modelu uzyskaną na zbiorze testowym.



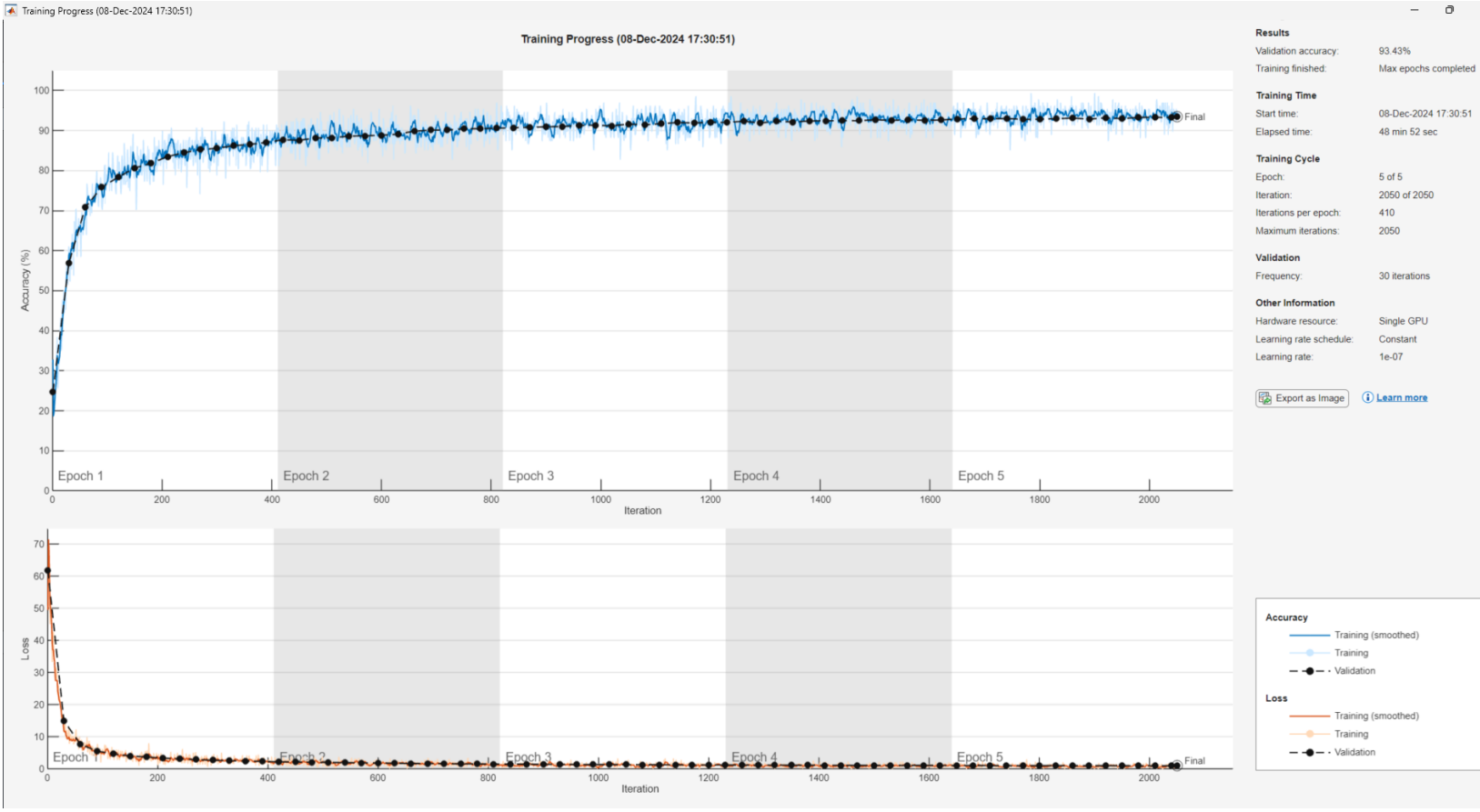
Rysunek 3: Wykres dla Ilr=1e-04, dokładność: 94.92%.



Rysunek 4: Wykres dla Ilr=1e-05, dokładność: 96.72%.



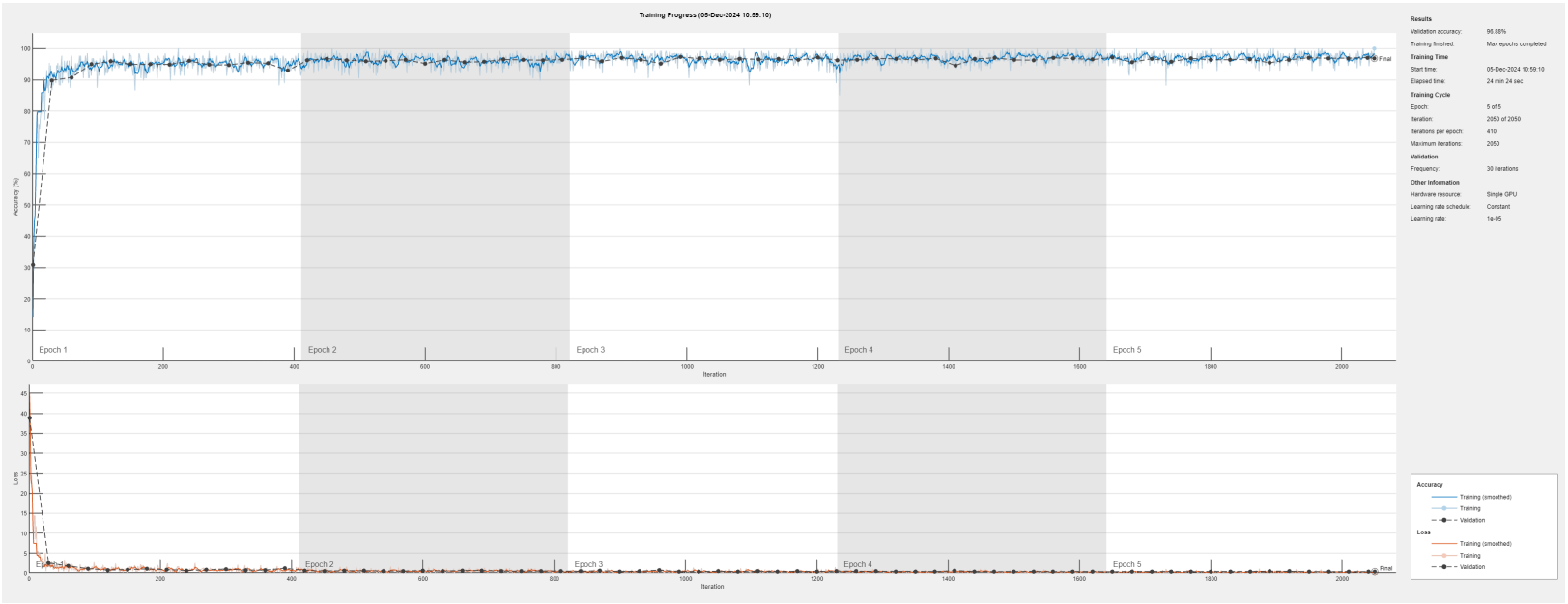
Rysunek 5: Wykres dla Ilr=1e-06, dokładność: 95.83%.



Rysunek 6: Wykres dla Ilr=1e-07, dokładność: 93.12%.

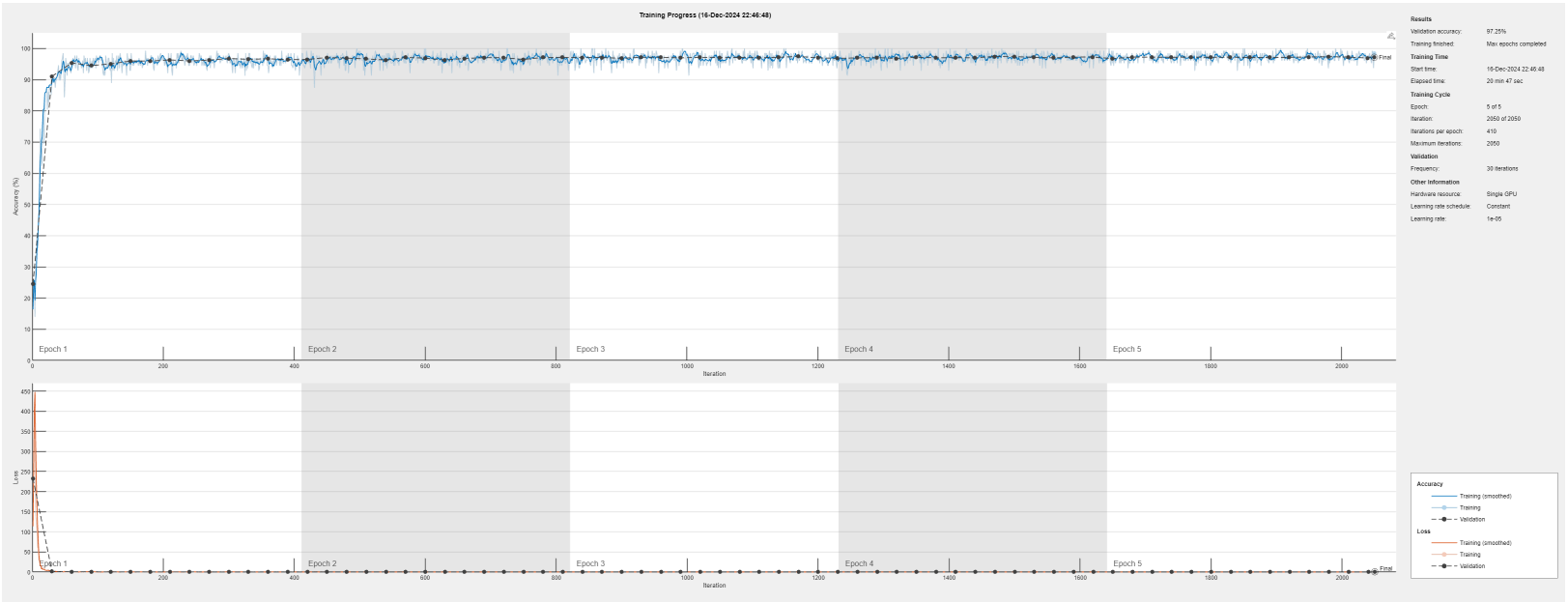
Najwyższą dokładność uzyskano na wykresie 4, gdzie współczynnik uczenia wynosił 10^{-5}.

W kolejnym etapie testowano różne konfiguracje warstw dla tych samym ustawień treningowych sieci.

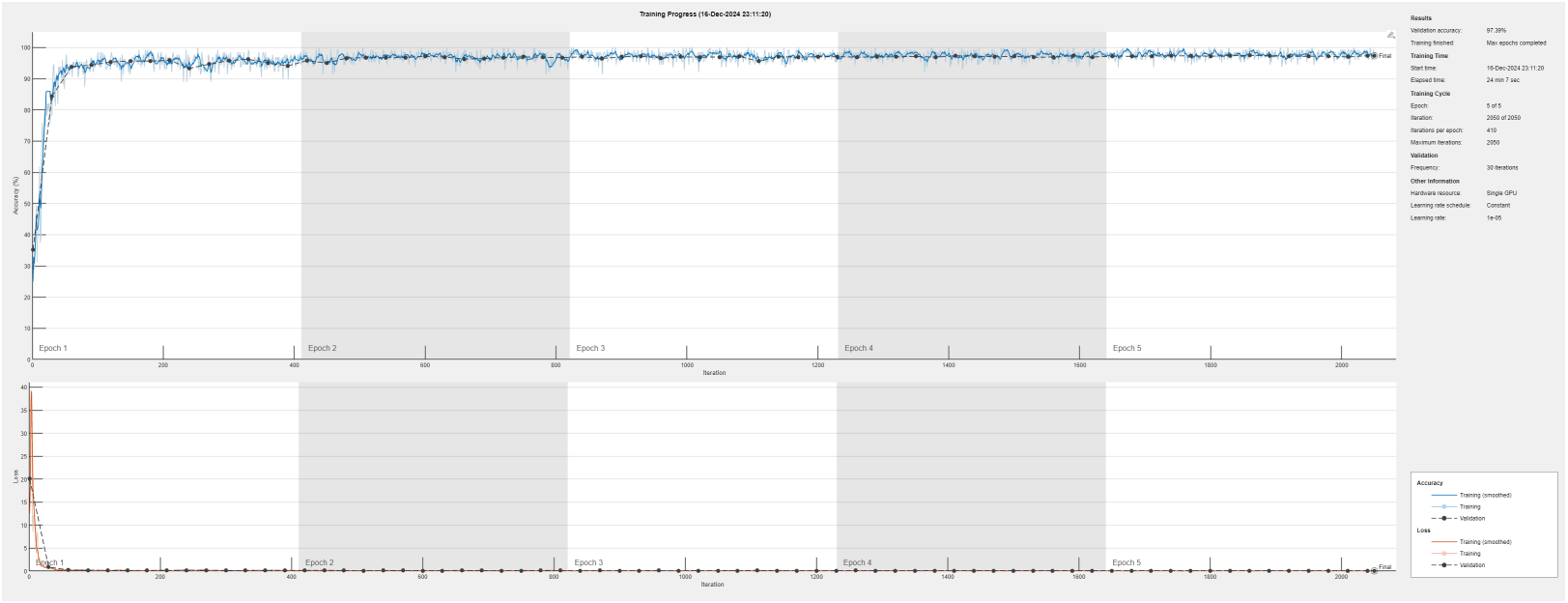


Rysunek 7: Wykres dla 4 warstw Fcl (1024, 512, 256, 5), dokładność: 96.98%.

Rysunek 8: Wykres dla 4 warstw Fcl (1024, 512, 256, 5) i Conv, dokładność: 93.40%.

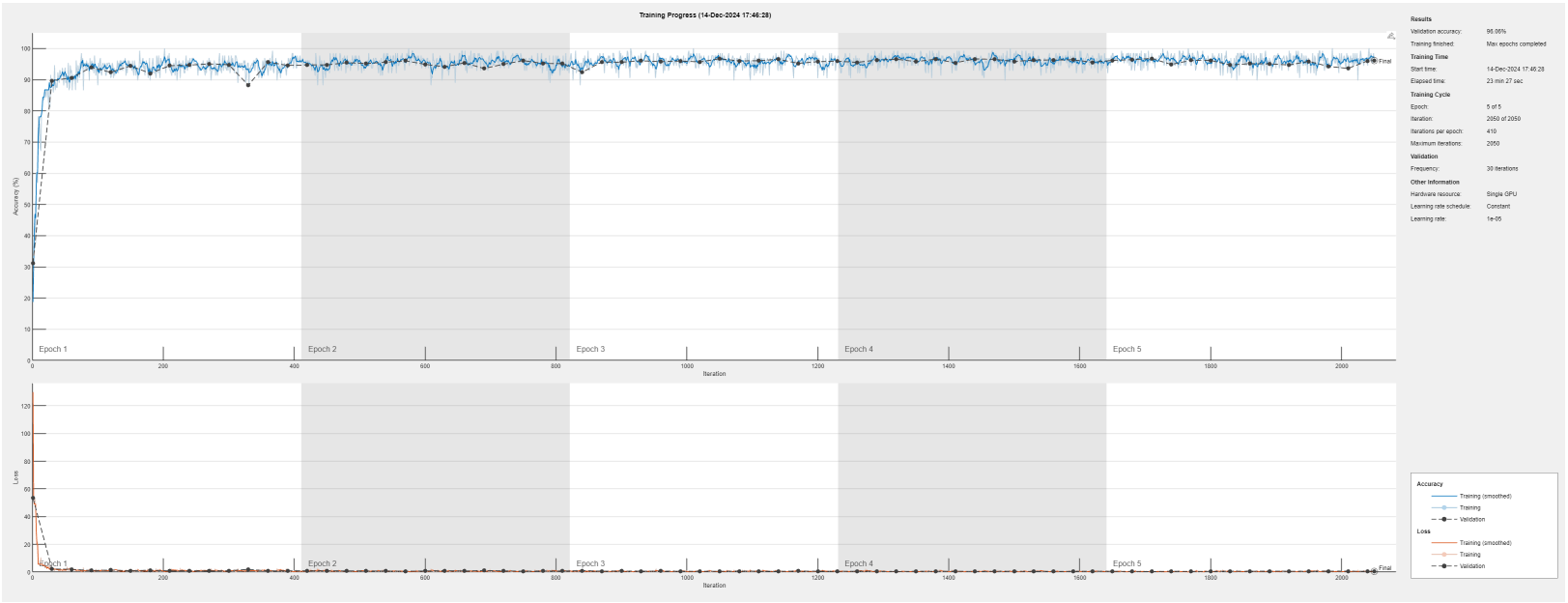


Rysunek 9: Wykres dla 2 warstw Fcl (256, 5), dokładność: 96.82%.



Rysunek 10: Wykres dla 2 warstw Fcl (256, 5) oraz Conv, dokładność: 97.44%.

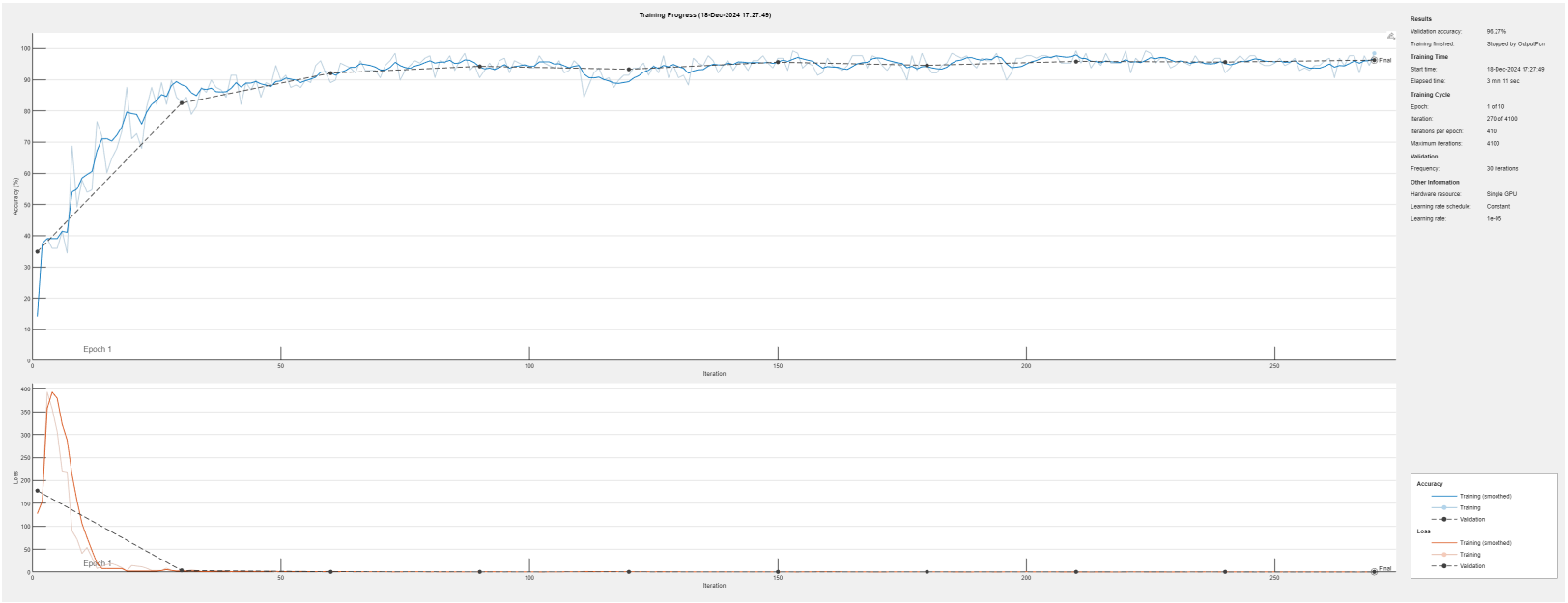
Zmieniono ustawienia treningowe sieci na tryb *adam* (Adaptive Moment Estimation) oraz by porównać z trybem *sgdm*, użyto 3 warstw w pełni połączonych (512, 256, 5) oraz współczynnika uczenia 10−5 tak jak przy testach na *Rysunek 4*.



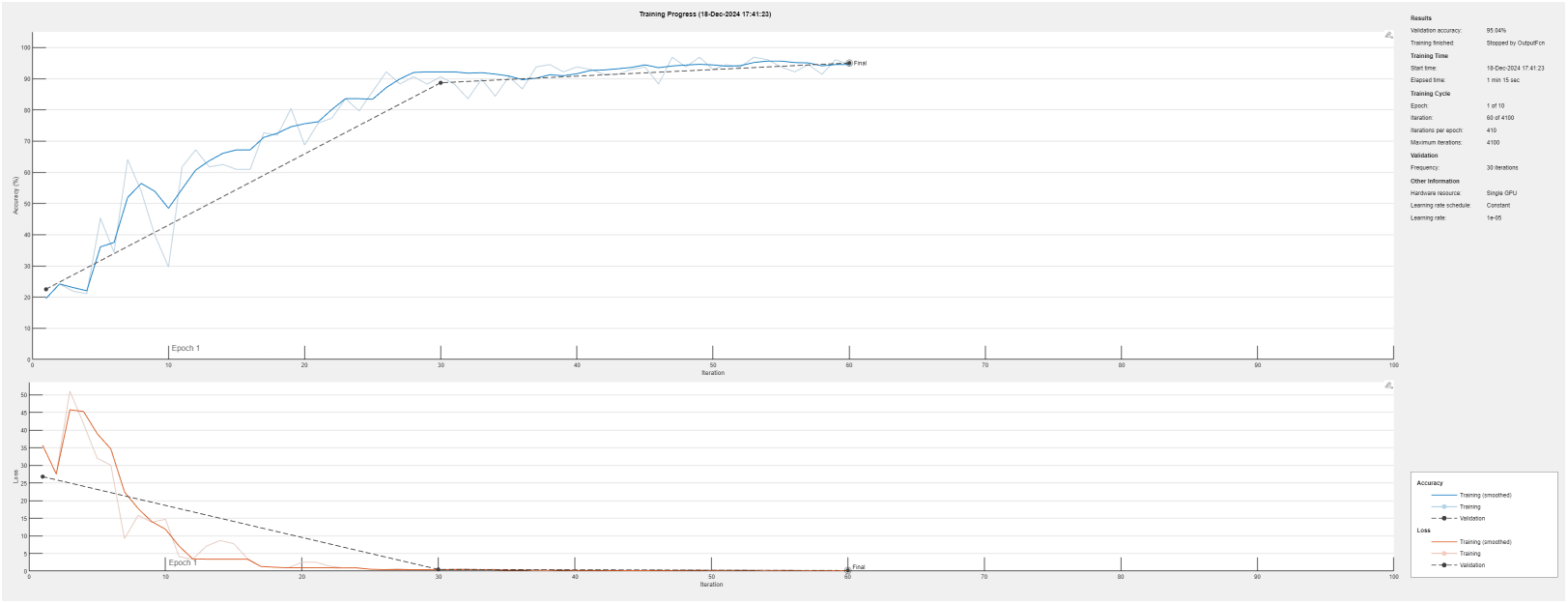
Rysunek 11: Wykres tryb *adam* dla 3 warstw Fcl (512, 256, 5), dokładność: 96.24%.

Oba tryby osiągnęły bardzo podobne dokładności *sgdm* - 96.72%, *adam* - 96.24%, przy czym *sgdm* był o 2 minuty szybszy.
Optymalizator *adam* cechuje się mniejszą stabilnością ponieważ dostosowuje współczynnik uczenia się dla każdego parametru, śledząc zarówno średnią wartość gradientów, jak i ich wariancję. *Sgdm* cechuje się większą stabilnością dzięki mechanizmowi momentu, który pozwala na wygładzenie zmian gradientu i przyspieszenie procesu uczenia, dodatkowo, w przeciwieństwie do Adama, współczynnik uczenia jest stały dla wszystkich parametrów, co sprawia, że optymalizacja jest bardziej przewidywalna. Analizując wykresy można zauważyć, że w epoce 5 na *Rysunek 10* widać wyraźniejsze wahania dokładności niż w *Rysunek 4*.

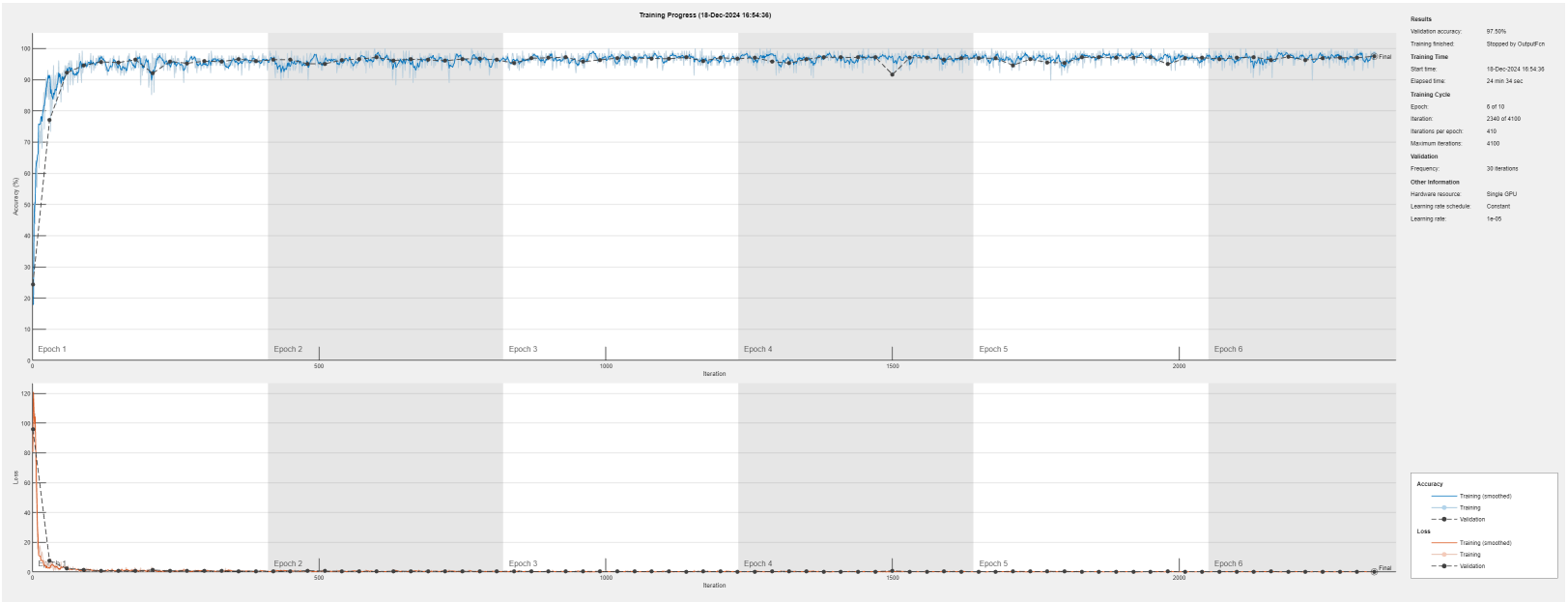
Do ustawień treningowych dodano funkcję stopu oraz wbudowaną opcję *ValidationPatience*, w celu określenia ilości epok po jakiej model przestaje wykazywać znaczący postęp w optymalizacji. W związku z tym, że używana jest warstwa *softmaxLayer* oraz *classificationLayer*, funkcją strat jest cross-entropia, która jest przeznaczona do problemów klasyfikacyjnych. Funkcja stopy zatrzymuje trening gdy sieć osiągnęła minimalne starty podczas walidacji wynoszące 0.25 (threshold), opcja ValidationPatience określa ile razy wartość strat walidacji może być mniejsza lub równa najlepszej poprzedniej wartości nim trening zostanie zatrzymany, ustawiona na 8. Testy przeprowdzono dla trzech najlepiej radzących sobie konfiguracji sieci - 2Fcl, 2Fcl i Conv, 3Fcl. Dla lepszej wizualizacji wyników stworzono macierz trafień.



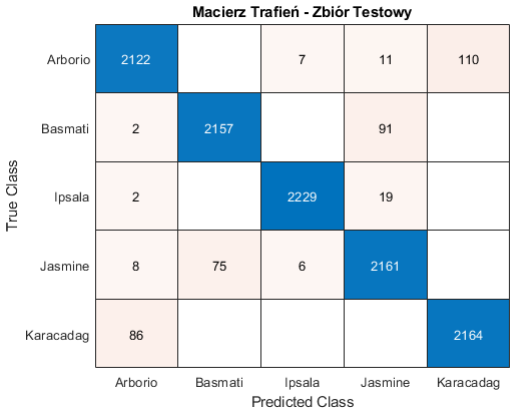
Rysunek 12: Wykres dla 2 warstw Fcl (256, 5), dokładność: 96.29%.



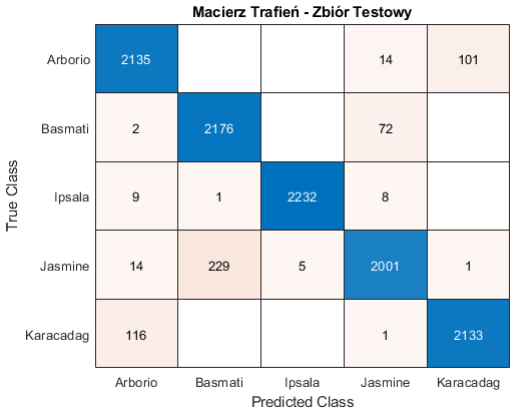
Rysunek 13: Wykres dla 2 warstw Fcl (256, 5) i Conv, dokładność: 94.90%.



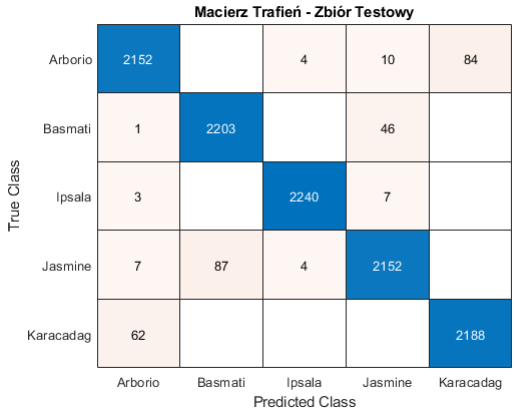
Rysunek 14: Wykres dla 3 warstw Fcl (512, 256, 5), dokładność: 97.20%.



(a) Dla *Rysunek 12*



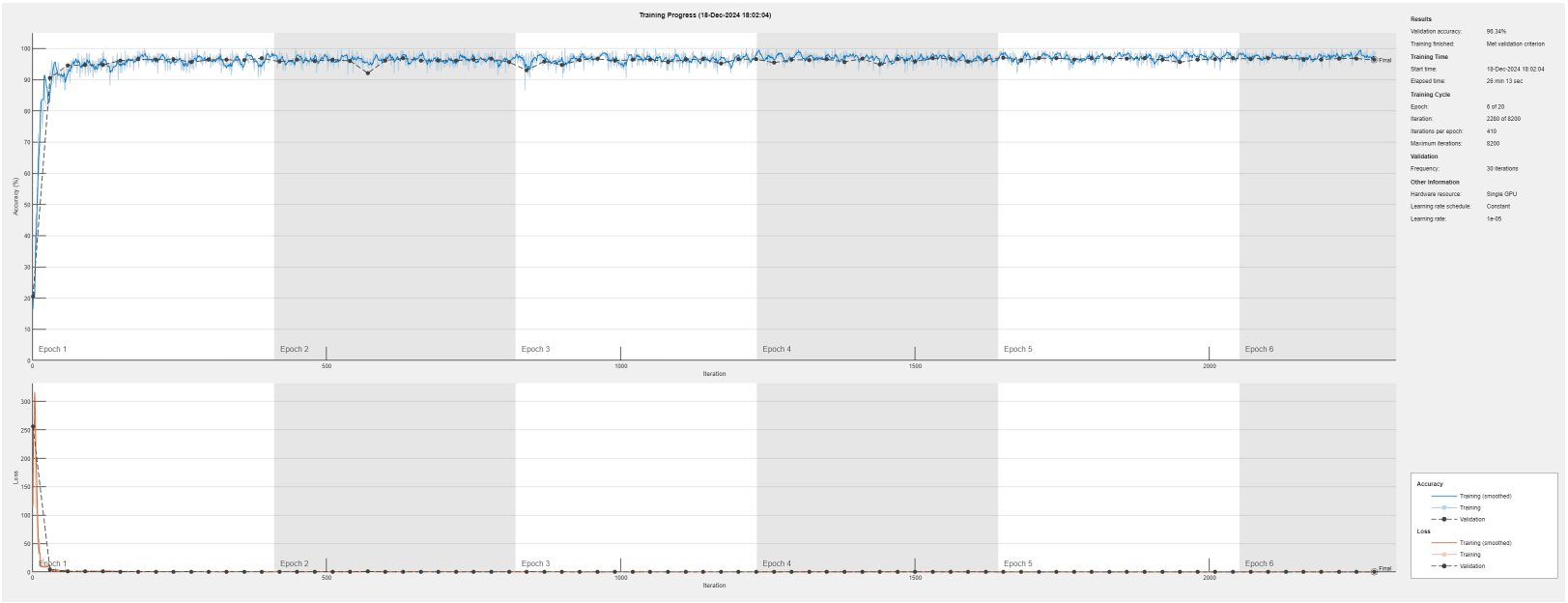
(b) Dla *Rysunek 13*



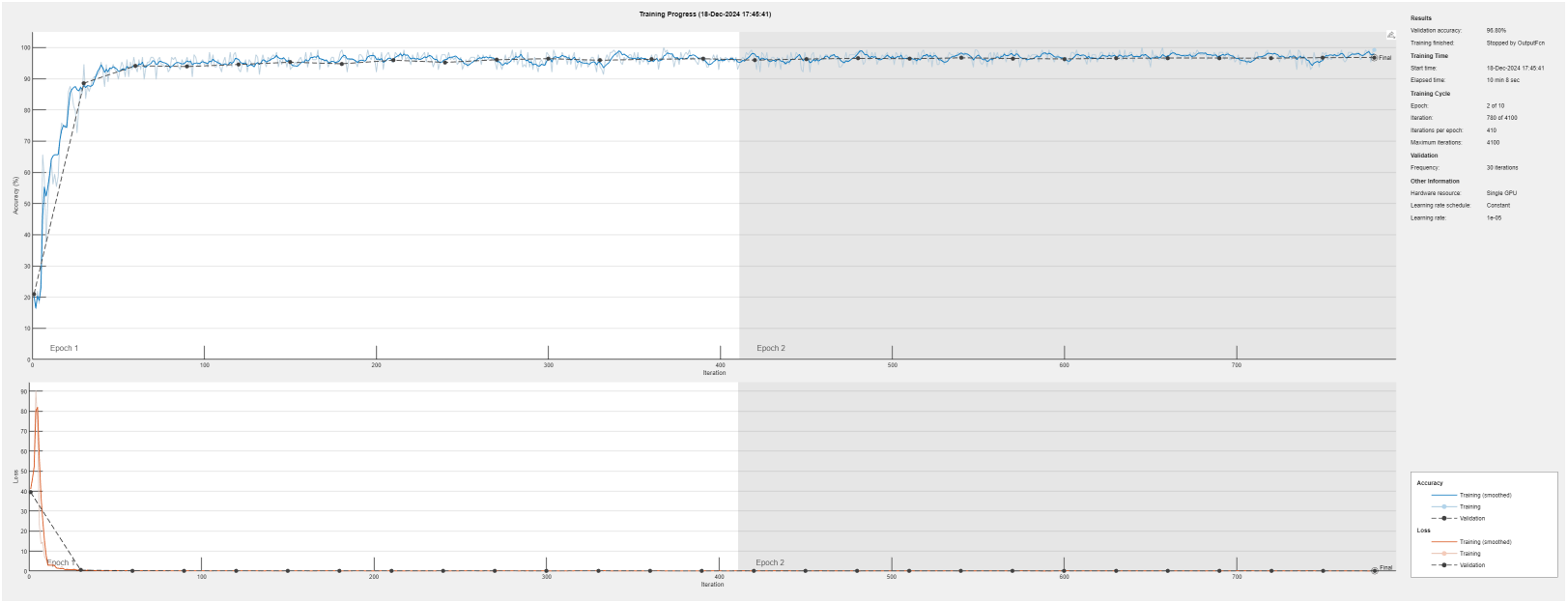
(c) Dla *Rysunek 14*

Rysunek 15: Macierze trafień

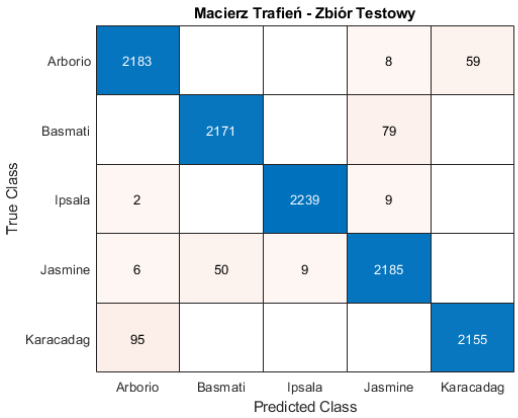
Threshold funkcji stop był za wysoki dla konfiguracji sieci z *Rysunek 12* oraz *Rysunek 13*, dlatego po dalszych obserwacjach wartości strat, zmniejszano jego wartość do 0.1.



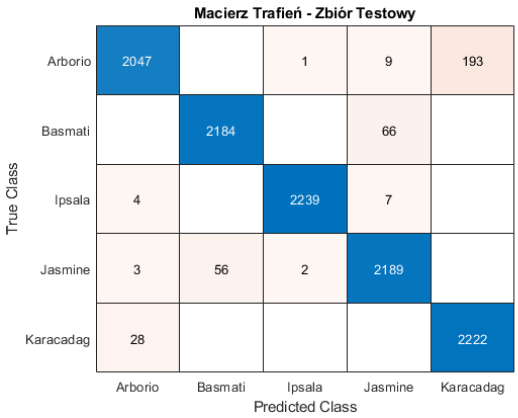
Rysunek 16: Wykres dla 2 warstw Fcl (256, 5), dokładność: 96.72%.



Rysunek 17: Wykres dla 2 warstw Fcl (256, 5) oraz Conv, dokładność: 97.18%.



(a) Dla *Rysunek 17*



(b) Dla *Rysunek 18*

Rysunek 18: Macierze trafień

4 Podsumowanie i wnioski

- Zmniejszenie współczynnika uczenia z 10^{-3} do 10^{-5} znacznie poprawiło wyniki.
- Dodanie przetasowania zbioru co każdą epokę pomogło zniwelować okresowość widoczną na *Rysunek 1.*
- Dodanie warstwy konwolucyjnej znacząco przyśpiesza progress sieci, straty szybciej maleją co wyraźnie widać porównując *Rysunek 16* oraz *Rysunek 17.* Wynika to prawdopodobnie nie z dodania samej warstwy konwolucyjnej, ale warstwy maxPooling2dLayer, która zmniejsza obraz, dzięki czemu sieć skupia się na analizie ziarenka, a nie dodatkowo tła.
- Format obrazów może znacząco uprościć proces nauczania sieci, wybrana baza pozwoliła potraktować obrazy jak dane czysto liczbowe.
- Sieć już w drugiej epoce osiąga zadowalające wyniki
- Zmiana trybu z *sgdm* na *adam* nie dokonała dużych zmian. Zastosowanie Adama i obserwacja wyników w pierwszych testach sieci może pozwolić na wybranie najlepszego współczynnika uczenia.
- Najlepszą konfiguracją sieci są dwie warstwy Fcl o liczbie neuronów 256 i 5 oraz jedna warstwa konwolucyjna z maxPooling2dLayer z współczynnik uczenia 10^{-5}.
- Odpowiednie dostosowanie threshold dla funkcji zatrzymującej pozwolić uniknąć przeuczenia, jednak zbyt wysoka wartość może wstrzymywać postęp. Zastosowanie opcji ValidationPatience zatrzymuje sieć w przypadku nie możliwości osiągniecia zadowalającej jakości, dzięki czemu nie marnujemy mocy obliczeniowej urządzenia.